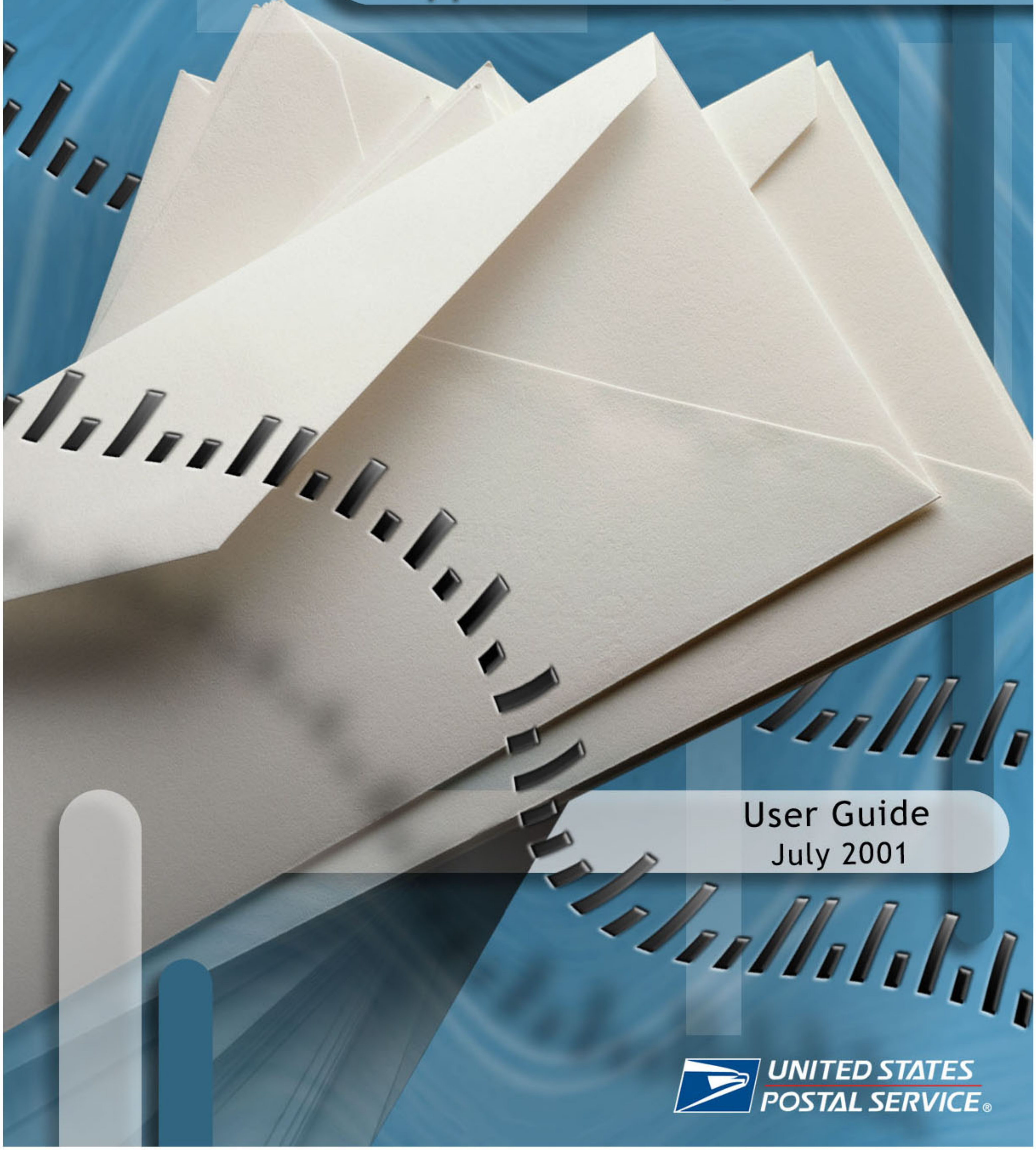
Address Matching System

Application Program Interface

User Guide

July 2001



UNITED STATES POSTAL SERVICE®

Table of Contents

Introduction

The USPS *Address Matching System Application Programming Interface User Guide* is the primary reference document for the USPS National Customer Support Center's Address Matching System product. The guide contains installation instructions for each platform as well as function descriptions.

The USPS Address Matching System is an application programming interface (API). As such, this guide should be used when the user wants to interface an application with the Address Matching System.

USPS Address Matching System Developer's Kit

The USPS Address Matching System Developer's Kit contains the following:

- API library(s) for each specific computer platform
- Interface definition file (ZIP4.H)
- Test utility (SAMPLE.EXE)
- Test utility source code
- Sample configuration data files
- User documentation

The test utility can be used to ensure that the Address Matching System and data files have been installed correctly and to provide access to our matching logic, which displays the standardized address returned by the matching engine. This enables you to verify the accuracy of the ZIP+4 results returned from your product.

CD-ROM Technical Support Department

If there are any question regarding the Address Matching System API, please call the United States Postal Service's National Customer Support Center, CD-ROM Technical Support Department at (800) 233-5866.

Installation Procedures for 32-Bit Windows (W32)

1. Create a directory on your hard drive in which to store the API files.

   ```
   Ex.  MD C:\AMS
   ```

2. Copy the Address Matching System files to your hard drive using the decryption program DEV_W32.EXE located on the CD-ROM in the DEV_KITS directory.

   ```
   Ex.  DEV_W32 CUST_ID OUTPUT_PATH PRODUCT_FILE
   ```

Note: *A customer ID (CUST_ID) should be obtained from CD-ROM Technical Support. The ID must be entered in uppercase letters.*

   The installation program must be executed from within the CD-ROM directory. This step needs to be performed once for each file listed in the file description in step 7 on the next page. Following initial installation, the only files that need to be installed with subsequent CD-ROM updates are the header files and libraries. A batch file is recommended to simplify this install process.

Note: *The customer ID provided by CD-ROM Technical Support is a generic ID subject to change at any time. DO NOT hard code this ID into an install program. For program installation, you may obtain a personal customer ID, which will not change for the duration of your Address Matching System API license, from CD-ROM Technical Support.*

   A. OUTPUT_PATH is the directory created in step 1.

   B. PRODUCT_FILE is the file from the list in step 7. This should not include any directory paths.

3. Change the directory locations in the Z4CONFIG.DAT file to the locations of the Address Matching System data files. (See Appendix C for a description of this file layout.)

   Example file for W32:

   ```
   APPLICATION        OTHER - ZIP+4
   COMPUTER           OTHER
   ADDRESS1           D:\AMSDATA\
   ADDRESS2
   ADDRESS3
   ADDRINDEX          D:\AMSDATA\
   CDROM
   CITYSTATE          D:\AMSDATA\
   CROSSREF           D:\AMSDATA\
   SYSTEM             C:\AMS\
   TABLE
   USER
   ADDR1SIZE
   ADDR2SIZE
   ADDR3SIZE
   ```

4. Run SAMPLE.EXE to test AMS.

5. Use SAMPLE.C as an example to create your own API application.

6. Refer to Section 3, API Functions, to test other API function commands.•••

7. The following is an explanation of the API files for W32:

a.	ZIP4_W32.DLL	ZIP4 dynamic-link library
b.	ZIP4_W32.LIB	Stub library to link with the user application
c.	ZIP4.H	Interface header file
d.	Z4CONFIG.DAT	File location file
e.	Z4CSYSDT.DAT	Date time file
f.	Z4CXLOG.DAT	Date time file
g.	SAMPLE.C	Sample C source file
h.	SAMPLE.EXE	Sample executable

Special Notes for W32

The 32-bit Windows version of the Address Matching System DLL was built with all export functions having the '_cdecl' calling convention, which has caused problems with some programming languages that do not support this convention. To provide access to the address matching routines in the DLL for non C and C++ languages, the DLL now contains a set of routines with the proper DLL calling convention '_stdcall.' These routines have separate names from the original routines to preserve linkage with existing programs, and the new names are a concatenation of the original function name and 'STD,' which implies the _stdcall calling convention, e.g.,

_cdecl function name	**_stdcall function name**
z4open()	z4openSTD()
z4adrinq()	z4adrinqSTD()
z4close()	z4closeSTD()

All of the _stdcall functions map directly to the original functions, so there is no loss in functionality. All existing functions have an associated _stdcall version, and all future additions to the DLL will contain both a _cdecl version and a _stdcall version.

Installation Procedures for OS/2

1. Create a directory on your hard drive in which to store the API files.

   ```
   Ex.  MD C:\AMS
   ```

2. Copy the Address Matching System files to your hard drive using the decryption program DEV_OS2.EXE located on the CD-ROM in the DEV_KITS directory.

   ```
   Ex.  DEV_OS2 CUST_ID OUTPUT_PATH PRODUCT_FILE
   ```

Note: *A customer ID (CUST_ID) should be obtained from CD-ROM Technical Support The ID must be entered in uppercase letters.*

   The installation program must be executed from within the CD-ROM directory one time for each file listed in the file description in step 7 on the next page. Following initial installation, the only files that need to be installed with subsequent CD-ROM updates are the header files and libraries. A command file is recommended to simplify this install process.

Note: *The customer ID provided by CD-ROM Technical Support is a generic ID subject to change at any time. DO NOT hard code this ID into an install program. For program installations, you may obtain a personal customer ID, which will not change for the duration of your Address Matching System API licence, from CD-ROM Technical Support.*

   A. OUTPUT_PATH is the directory created in step 1.

   B. PRODUCT_FILE is the file from the list in step 7. This should not include any directory paths.

3. Change the directory locations in the Z4CONFIG.DAT file to the locations of the Address Matching System data files. (See Appendix C for a description of this file layout.)

   Example file for OS/2:

   ```
   APPLICATION        OTHER - ZIP+4
   COMPUTER           OTHER
   ADDRESS1           D:\AMSDATA\
   ADDRESS2
   ADDRESS3
   ADDRINDEX          D:\AMSDATA\
   CDROM
   CITYSTATE          D:\AMSDATA\
   CROSSREF           D:\AMSDATA\
   SYSTEM             C:\AMS\
   TABLE
   USER
   ADDR1SIZE
   ADDR2SIZE
   ADDR3SIZE
   ```

4. Run SAMPLE.EXE to test the Address Matching System.

5. Use SAMPLE.C as an example to create your own API application.

6. Refer to Section 3, API Functions, to test other API function commands.•••

7. The following is an explanation of the API files for OS/2:

a.	ZIP4_OS2.DLL	ZIP4 dynamic-link library
b.	ZIP4_OS2.LIB	Stub library to link with the user application
c.	ZIP4.H	Interface header file
d.	Z4CONFIG.DAT	File location file
e.	Z4CSYSDT.DAT	Date time file
f.	Z4CXLOG.DAT	Date time file
g.	SAMPLE.C	Sample C source file
h.	SAMPLE.EXE	Sample executable

Installation Procedures for Macintosh (MAC)

1. Create a folder on your hard drive in which to store the API files by selecting **File** > **New Folder** from the desktop pull-down menu or pressing COMMAND+N.

2. Copy the Address Matching System files to your hard drive using the decryption program DEV_MAC.

***Note**: The installation program is provided on diskette because the ISO9660 file-system format of the Address Matching System CD-ROM cannot support the Macintosh executable file format.*

To prepare the installation program, complete the following steps:

   A. Copy the installation program from the diskette to the folder created in step 1.

   B. Copy the MAC folder (directory) and DEVKITS.TBL from the DEV_KITS directory to the folder created in step 1.

3. Run the DEV_MAC installation program from the folder created in step 1.

   A. Select **Install AMS** from the File menu.

   B. Enter the customer ID obtained from CD-ROM Technical Support. The ID must be entered in uppercase letters.

   C. Enter the full destination path to the folder created in step 1.

   D. Ensure that the check-boxes for all of the files are selected for initial installation: ZIP4_MAC.LIB, SAMPLE.C, Z4CXLOG.DAT, Z4CSYSDT.DAT, Z4CONFIG.DAT, ZIP4.H

4. Change the directory locations in the Z4CONFIG.DAT file to the locations of the Address Matching System data files. The delimiter ":" is used to separate the folder names in the Macintosh hierarchical folder system. The device volume name must appear at the beginning of each applicable listing, and a colon (:) must appear at the end of each listing. (See Appendix C for a description of this file layout.)

   Example file for MAC:

```
APPLICATION        OTHER - ZIP+4
COMPUTER           OTHER
ADDRESS1           USPS_97255:
ADDRESS2
ADDRESS3
ADDRINDEX          USPS_97255:
CDROM
CITYSTATE          USPS_97255:
CROSSREF           USPS_97255:
SYSTEM             My_Hd:AMS_Folder:
TABLE
USER
ADDR1SIZE
ADDR2SIZE
ADDR3SIZE
```

5. Use SAMPLE.C to create your own API application.

6. Refer to Section 3, API Functions, to test other API applications.

7. The following is an explanation of the API files for MAC:

a.	ZIP4_MAC.LIB	Static-link library
b.	SAMPLE.C	Sample C source file
c.	Z4CXLOG.DAT	Date time file
d.	Z4CSYSDT.DAT	File location file
e.	SAMPLE	Sample executable
f.	ZIP4.H	Interface header file

Note: *When performing CD-ROM updates of the Address Matching System API library, follow steps 2B through 3D. When selecting the files to update in 3D, select only* ZIP4_MAC.LIB *and* ZIP4.H. *These are the only two files that will periodically change in the Macintosh Developer's Kit.*

Installation Procedures for SCO Unix

1. Create a directory on your hard drive in which to store the API files.

   ```
   Ex.  mkdir /usr/src/ams
   ```

2. Copy the Address Matching System files to your hard drive using the decryption program DEV_SCO.EXE located on the CD-ROM in the DEV_KITS directory.

   ```
   Ex.  DEV_SCO CUST_ID OUTPUT_PATH PRODUCT_FILE
   ```

Note: *A customer ID (CUST_ID) should be obtained from CD-ROM Technical Support. The ID must be entered in uppercase letters.*

The installation program must be executed from within the CD-ROM directory one time for each file listed in the file description in step 7 on the next page. Following initial installation, the only files that need to be installed with subsequent CD-ROM updates are the header files and libraries. A shell script is recommended to simplify the install process.

Note: *The customer ID provided by CD-ROM Technical Support is a generic ID subject to change at any time. DO NOT hard code this ID into an install program. For program installations, you may obtain a personal customer ID, which will not change for the duration of your Address Matching System API licences, from CD-ROM Technical Support.*

   A. OUTPUT_PATH is the directory created in step 1.

   B. PRODUCT_FILE is the file from the list in step 7. This should not include any directory paths.

3. Change the directory locations in the Z4CONFIG.DAT file to the locations of the Address Matching System data files. (See Appendix C for a description of this file layout.)

   Example file for SCO:

   ```
   APPLICATION        OTHER - ZIP+4
   COMPUTER           OTHER
   ADDRESS1           /mount/cdrom/
   ADDRESS2
   ADDRESS3
   ADDRINDEX          /mount/cdrom/
   CDROM
   CITYSTATE          /mount/cdrom/
   CROSSREF           /mount/cdrom/
   SYSTEM             /usr/src/ams/
   TABLE
   USER
   ADDR1SIZE
   ADDR2SIZE
   ADDR3SIZE
   ```

4. Run SAMPLE.EXE to test the Address Matching System.

   A. CHMOD on SAMPLE.EXE to rwx.

   B. CHMOD on Z4CXLOG.DAT to rw.

5. Use SAMPLE.C as an example to create your own API application.

6. Refer to Section 3, API Functions, to test other API function commands.

7. The following is an explanation of the API files for SCO:

a.	LIBZ4SCO.SO	ZIP4 shared library
b.	ZIP4_SCO.A	Static-link library (not recommended)
c.	ZIP4.H	Interface header file
d.	Z4CONFIG.DAT	File location file
e.	Z4CSYSDT.DAT	Date time file
f.	Z4CXLOG.DAT	Date time file
g.	SAMPLE.C	Sample C source file
h.	SAMPLE.EXE	Sample executable

Special Notes for SCO Users

The Address Matching System CD-ROM uses the ISO9660 file-system format, which stores file names in uppercase letters with a version control number appended to the end. However, the API requires that the CD-ROM file names appear in lowercase letters without the version number. Some versions of Unix will automatically accommodate file name conversion during the mount process, but some require the user to specify this conversion explicitly with the options of the “mount” command. Please see the **man** pages on mount for more information on these options.

The Address Matching System SCO API development kit contains both a static-link and a shared library. The static-link library is provided for compatibility with older programs written before the shared library was available. The USPS does not recommend use of the static-link library because logic changes are often made to the API, and the user would have to re-link the executable file with the static-link library every time there is an update. Also, in compliance with CASS rules, the API code is set to expire at the end of the current CASS cycle. If this date is reached without re-linking with a newer API, a user’s application will stop functioning.

To avoid these problems, the USPS recommends using the shared library so that applications can gain immediate access to any logic changes simply by installing the new shared library. User applications do not need to be re-linked when a new shared library is provided on CD-ROM updates.

Installation Procedures for Sun Unix

1. Create a directory on your hard drive in which to store the API files.

   ```
   Ex.  mkdir /usr/src/ams
   ```

2. Copy the Address Matching System files to your hard drive using the decryption program DEV_SUN.EXE located on the CD-ROM in the DEV_KITS directory.

   ```
   Ex.  DEV_SUN.EXE CUST_ID OUTPUT_PATH PRODUCT_FILE
   ```

Note: *A customer ID (CUST_ID) should be obtained from CD-ROM Technical Support The ID must be entered in uppercase letters.*

   The installation program must be executed from within the CD-ROM directory one time for each file listed in the file description in step 7 on the next page. Following initial installation, the only files that need to be installed with subsequent CD-ROM updates are the header files and libraries. A shell script is recommended to simplify the install process.

Note: *The customer ID given out by CD-ROM Technical Support is a generic ID subject to change at any time. DO NOT hard code this ID into an install program. For program installations, you may obtain a personal customer ID, which will not change for the duration of your Address Matching System API license, from CD-ROM Technical Support.*

   A. OUTPUT_PATH is the directory created in step 1.

   B. PRODUCT_FILE is the file from the list in step 7. This should not include any directory paths.

3. Change the directory locations in the Z4CONFIG.DAT file to the locations of the Address Matching System data files. (See Appendix C for a description of this file layout.)

   Example file for SUN:

   ```
   APPLICATION       OTHER - ZIP+4
   COMPUTER          OTHER
   ADDRESS1          /mount/cdrom/
   ADDRESS2
   ADDRESS3
   ADDRINDEX         /mount/cdrom/
   CDROM
   CITYSTATE         /mount/cdrom/
   CROSSREF          /mount/cdrom/
   SYSTEM            /usr/src/ams/
   TABLE
   USER
   ADDR1SIZE
   ADDR2SIZE
   ADDR3SIZE
   ```

4. Run SAMPLE.EXE to test the Address Matching System.

   A. CHMOD on SAMPLE.EXE to rwx.

   B. CHMOD on Z4CXLOG.DAT to rw.

5. Use SAMPLE.C as an example to create your own API application.

6. Refer to Section 3, API Functions, to test other API function calls.
7. The following is an explanation of the API files for SUN:

a.	LIBZ4SUN.SO	ZIP4 shared library
b.	ZIP4_SUN.A	Static link library; not recommended
c.	ZIP4.H	Interface header file
d.	Z4CONFIG.DAT	File location file
e.	Z4CSYSDT.DAT	Date time file
f.	Z4CXLOG.DAT	Date time file
g.	SAMPLE.C	Sample C source file
h.	SAMPLE.EXE	Sample executable

Special Notes for SUN Users

The Address Matching System CD-ROM uses the ISO9660 file-system format, which stores file names in uppercase letters with a version control number appended to the end. The API requires that the CD-ROM file names appear in lowercase letters without the version number. Some versions of Unix will automatically accommodate file name conversion during the mount process, but some require the user to specify the conversion explicitly with the options of the “mount” command. Please see the **man** pages on mount for more information on these options.

The Address Matching System SUN API Developer’s Kit contains both a static-link and a shared library. The static-link library is provided for compatibility with older programs written before the shared library was available. The USPS does not recommend use of the static-link library because logic changes are often made to the API, and the user would have to re-link the executable file with the AMS static-link library every time there is an update. Also, in compliance with CASS rules, the API code is set to expire at the end of the current CASS cycle. If this date is reached without re-linking with a newer API, a user’s application will stop functioning.

To avoid these problems the USPS recommends using the AMS shared library so that user applications can gain immediate access to any logic changes simply by installing the new shared library. User applications do not need to be re-linked when a new shared library is provided on CD-ROM updates.

Installation Procedures for HP Unix

1. Create a directory on your hard drive in which to store the API files.

   ```
   Ex.  mkdir /usr/src/ams
   ```

2. Copy the Address Matching System files to your hard drive using the decryption program DEV_HP.EXE located on the CD-ROM in the DEV_KITS directory.

   ```
   Ex.  DEV_HP.EXE CUST_ID OUTPUT_PATH PRODUCT_FILE
   ```

Note: *A customer ID (CUST_ID) should be obtained from CD-ROM Technical Support. The ID must be entered in uppercase letters.*

   The installation program must be executed from within the CD-ROM directory one time for each file listed in the file description in step 7 on the next page. Following initial installation, the only files that need to be installed with subsequent CD-ROM updates are the header files and libraries. A shell script is recommended to simplify this install process.

Note: *The customer ID provided by CD-ROM Technical Support is a generic ID subject to change at any time. DO NOT hard code this ID into an install program. For program installations, you may obtain a personal customer ID, which will not change for the duration of your Address Matching System API license, from CD-ROM Technical Support.*

   A. OUTPUT_PATH is the directory created in step 1.

   B. PRODUCT_FILE is the file from the list in step 7. This should not include any directory paths.

3. Change the directory locations in the Z4CONFIG.DAT file to the locations of the Address Matching System data files. (See Appendix C for a description of this file layout.)

   Example file for HP:

   ```
   APPLICATION        OTHER - ZIP+4
   COMPUTER           OTHER
   ADDRESS1           /mount/cdrom/
   ADDRESS2
   ADDRESS3
   ADDRINDEX          /mount/cdrom/
   CDROM
   CITYSTATE          /mount/cdrom/
   CROSSREF           /mount/cdrom/
   SYSTEM             /usr/src/ams/
   TABLE
   USER
   ADDR1SIZE
   ADDR2SIZE
   ADDR3SIZE
   ```

4. Run SAMPLE.EXE to test Address Matching System.

   A. CHMOD on SAMPLE.EXE to rwx.

   B. CHMOD on Z4CXLOG.DAT to rw.

5. Use SAMPLE.C as an example to create your own API application.

6. Refer to Section 3, API Functions, to test other API function commands.

7. The following is an explanation of the API files for HP Unix:

a.	LIBZ4HP.SL	ZIP4 shared library
b.	ZIP4_HP.A	Static-link library (not recommended)
c.	ZIP4.H	Interface header file
d.	Z4CONFIG.DAT	File location file
e.	Z4CSYSDT.DAT	Date time file
f.	Z4CXLOG.DAT	Date time file
g.	SAMPLE.C	Sample C source file
h.	SAMPLE.EXE	Sample executable

Special Notes for HP Users

The Address Matching System CD-ROM uses the ISO9660 file-system format, which stores file names in uppercase letters with a version control number appended to the end. However, the API requires that the CD-ROM file names appear in lowercase letters without the version number. Some versions of Unix will automatically accommodate file-name conversion during the mount process, but some require the user to specify the conversion explicitly with the options of the “mount” command. Please see the **man** pages on mount for more information on these options.

On some HP systems, it may be necessary to use the Portable File System (pfs) class of mount commands to properly mount the CD-ROM. This may require starting the pfsd daemon at system boot time. See the **man** pages on pfs, pfs_mount, and pfsd for more help.

The Address Matching System HP API Developer’s Kit contains both a static-link and a shared library. The static-link library is provided for compatibility with older programs written before the shared library was available. The USPS does not recommend use of the static-link library because logic changes are often made to the API, and the user would have to re-link the executable file with the static-link library every time there is an update. Also, in compliance with CASS rules, the API code is set to expire at the end of the current CASS cycle. If this date is reached without re-linking with a newer API, a user’s application will stop functioning.

To avoid these problems the USPS recommends using the shared library so that user applications can gain immediate access to any logic changes simply by installing the new shared library. User applications do not need to be re-linked when a new shared library is provided on CD-ROM updates.

Installation Procedures for Digital Unix (OSF)

1. Create a directory on your hard drive in which to store the API files.

   ```
   Ex.  mkdir  /usr/src/ams
   ```

2. Copy the DEV_OSF.EXE located on the CD-ROM in the DEV_KITS directory.

   ```
   Ex.  DEV_OSF.EXE CUST_ID OUTPUT_PATH PRODUCT_FILE
   ```

Note: *A customer ID (CUST_ID) should be obtained from CD-ROM Technical Support. The ID must be entered in uppercase letters.*

The installation program must be executed from within the CD-ROM directory one time for each file listed in the file description in step 7 on the next page. Following initial installation, the only files that need to be installed with subsequent CD-ROM updates are the header files and libraries. A shell script is recommended to simplify this install process.

Note: *The customer ID provided by CD-ROM Technical Support is a generic ID subject to change at any time. DO NOT hard code this ID into an install program. For program installations, you may obtain a personal customer ID, which will not change for the duration of your Address Matching System API license, from CD-ROM Technical Support.*

A. OUTPUT_PATH is the directory created in step 1.

B. PRODUCT_FILE is the file from the list in step 7. This should not include any directory paths.

3. Change the directory locations in the Z4CONFIG.DAT file to the locations of the Address Matching System data files. (See Appendix C for a description of this file layout.)

   Example file for OSF:

   ```
   APPLICATION        OTHER - ZIP+4
   COMPUTER           OTHER
   ADDRESS1           /mount/cdrom/
   ADDRESS2
   ADDRESS3
   ADDRINDEX          /mount/cdrom/
   CDROM
   CITYSTATE          /mount/cdrom/
   CROSSREF           /mount/cdrom/
   SYSTEM             /usr/src/ams/
   TABLE
   USER
   ADDR1SIZE
   ADDR2SIZE
   ADDR3SIZE
   ```

4. Run SAMPLE.EXE to test Address Matching System.

   A. CHMOD on SAMPLE.EXE to rwx.

   B. CHMOD on Z4CXLOG.DAT to rw.

5. Use SAMPLE.C as an example to create your own API application.

6. Refer to Section 3, API Functions, to test other API function commands.

7. The following is an explanation of the API files for AIX Unix:

a.	LIBZ4OSF.SO	ZIP4 shared library
b.	ZIP4.H	Interface header file
c.	Z4CONFIG.DAT	File location file
d.	Z4CSYSDT.DAT	Date time file
e.	Z4CXLOG.DAT	Date time file
f.	SAMPLE.C	Sample C source file
g.	SAMPLE.EXE	Sample executable

Special Notes for Digital Unix Users

The Address Matching System CD-ROM uses the ISO9660 file-system format, which stores file names in uppercase letters with a version control number appended to the end. However, the API requires that the CD-ROM file names appear in lowercase letters without the version number. Some versions of Unix will automatically accommodate file-name conversion during the mount process, but some require the user to specify the conversion explicitly with the options of the “mount” command. Please see the **man** pages on mount for more information on these options.

Installation Procedures for IBM AIX Unix

1. Create a directory on your hard drive in which to store the API files.

   ```
   Ex.  mkdir  /usr/src/ams
   ```

2. Copy the Address Matching System files to your hard drive using the decryption program DEV_AIX.EXE located on the CD-ROM in the DEV_KITS directory.

   ```
   Ex.  DEV_AIX.EXE CUST_ID OUTPUT_PATH PRODUCT_FILE
   ```

Note: *A customer ID (CUST_ID) should be obtained from CD-ROM Technical Support. The ID must be entered in uppercase letters.*

   The installation program must be executed from within the CD-ROM directory one time for each file listed in the file description in step 7 on the next page. Following initial installation, the only files that need to be installed with subsequent CD-ROM updates are the header files and libraries. A shell script is recommended to simplify this install process.

Note: *The customer ID provided by CD-ROM Technical Support is a generic ID subject to change at any time. DO NOT hard code this ID into an install program. For program installations, you may obtain a personal customer ID, which will not change for the duration of your Address Matching System API license, from CD-ROM Technical Support.*

   A. OUTPUT_PATH is the directory created in step 1.

   B. PRODUCT_FILE is the file from the list in step 7. This should not include any directory paths.

3. Change the directory locations in the Z4CONFIG.DAT file to the locations of the Address Matching System data files. (See Appendix C for a description of this file layout.)

   Example file for AIX:

   ```
   APPLICATION        OTHER - ZIP+4
   COMPUTER           OTHER
   ADDRESS1           /mount/cdrom/
   ADDRESS2
   ADDRESS3
   ADDRINDEX          /mount/cdrom/
   CDROM
   CITYSTATE          /mount/cdrom/
   CROSSREF           /mount/cdrom/
   SYSTEM             /usr/src/ams/
   TABLE
   USER
   ADDR1SIZE
   ADDR2SIZE
   ADDR3SIZE
   ```

4. Run SAMPLE.EXE to test Address Matching System.

   A. CHMOD on SAMPLE.EXE to rwx.

   B. CHMOD on Z4CXLOG.DAT to rw.

5. Use SAMPLE.C as an example to create your own API application.

6. Refer to Section 3, API Functions, to test other API function commands.
7. The following is an explanation of the API files for AIX Unix:

a.	LIBZ4OSF.SO	ZIP4 shared library
b.	ZIP4.H	Interface header file
c.	Z4CONFIG.DAT	File location file
d.	Z4CSYSDT.DAT	Date time file
e.	Z4CXLOG.DAT	Date time file
f.	SAMPLE.C	Sample C source file
g.	SAMPLE.EXE	Sample executable

Special Notes for AIX Unix Users

The Address Matching System CD-ROM uses the ISO9660 file-system format, which stores file names in uppercase letters with a version control number appended to the end. However, the API requires that the CD-ROM file names appear in lowercase letters without the version number. Some versions of Unix will automatically accommodate file-name conversion during the mount process, but some require the user to specify the conversion explicitly with the options of the “mount” command. Please see the **man** pages on mount for more information on these options.

Installation Procedures for Linux

1. Create a directory on your hard drive in which to store the API files.

   ```
   Ex.  mkdir  /usr/src/ams
   ```

2. Copy the Address Matching System files to your hard drive using the decryption program DEV_LNX.EXE located on the CD-ROM in the DEV_KITS directory.

   ```
   Ex.  DEV_LNX.EXE CUST_ID OUTPUT_PATH PRODUCT_FILE
   ```

Note: *A customer ID (CUST_ID) should be obtained from CD-ROM Technical Support. The ID must be entered in uppercase letters.*

   The installation program must be executed from within the CD-ROM directory one time for each file listed in the file description in step 7 on the next page. Following initial installation, the only files that need to be installed with subsequent CD-ROM updates are the header files and libraries. A shell script is recommended to simplify this install process.

Note: *The customer ID provided by CD-ROM Technical Support is a generic ID subject to change at any time. DO NOT hard code this ID into an install program. For program installations, you may obtain a personal customer ID, which will not change for the duration of your Address Matching System API license, from CD-ROM Technical Support.*

   A. OUTPUT_PATH is the directory created in step 1.

   B. PRODUCT_FILE is the file from the list in step 7. This should not include any directory paths.

3. Change the directory locations in the Z4CONFIG.DAT file to the locations of the Address Matching System data files. (See Appendix C for a description of this file layout.)

   Example file for LNX:

   ```
   APPLICATION        OTHER - ZIP+4
   COMPUTER           OTHER
   ADDRESS1           /mount/cdrom/
   ADDRESS2
   ADDRESS3
   ADDRINDEX          /mount/cdrom/
   CDROM
   CITYSTATE          /mount/cdrom/
   CROSSREF           /mount/cdrom/
   SYSTEM             /usr/src/ams/
   TABLE
   USER
   ADDR1SIZE
   ADDR2SIZE
   ADDR3SIZE
   ```

4. Run SAMPLE.EXE to test Address Matching System.

   A. CHMOD on SAMPLE.EXE to rwx.

   B. CHMOD on Z4CXLOG.DAT to rw.

5. Use SAMPLE.C as an example to create your own API application.

6. Refer to Section 3, API Functions, to test other API function commands.
7. The following is an explanation of the API files for LNX:

a.	LIBZ4LNX.SO	ZIP4 shared library
b.	ZIP4_LNX	Static link library; not recommended
c.	ZIP4.H	Interface header file
d.	Z4CONFIG.DAT	File location file
e.	Z4CSYSDT.DAT	Date time file
f.	Z4CXLOG.DAT	Date time file
g.	SAMPLE.C	Sample C source file
h.	SAMPLE.EXE	Sample executable

Special Notes for Linux Users

The Address Matching System CD-ROM uses the ISO9660 file-system format, which stores file names in uppercase letters with a version control number appended to the end. However, the API requires that the CD-ROM file names appear in lowercase letters without the version number. Some versions of Unix will automatically accommodate file-name conversion during the mount process, but some require the user to specify the conversion explicitly with the options of the “mount” command. Please see the **man** pages on mount for more information on these options.

The Address Matching System LINUX API Developer’s Kit contains both a static-link and a shared library. The static-link library is provided for compatibility with older programs written before the shared library was available. The USPS does not recommend use of the static-link library because logic changes are often made to the API, and the user would have to re-link the executable files with the AMS static-link library every time there is an update. Also, in compliance with CASS rules, the API code is set to expire at the end of the current CASS cycle. If this date is reached without re-linking with a newer API, a user’s application will stop functioning.

To avoid these problems, the USPS recommends using the AMS shared library so that user applications can gain immediate access to any logic changes simply by installing the new shared library. User applications do not need to be re-linked when a new shared library is provided on CD-ROM updates.

Functions

The following functions are used to perform inquiries on addresses and 9-digit ZIP Codes:

• z4open()	Open the Address Matching System
• z4opencfg()	Open the Address Matching System with a Special Configuration File
• z4adrinq()	Address Inquiry
• z4adrkey()	Address Sort Key
• z4xrfinq()	9-digit Inquiry
• z4adrstd()	Address Standardization
• z4close()	Closing Address Matching System
• z4remove()	Removing the Address Matching System from memory
• z4ctyget()	Read City/State File by Key
• z4ctynxt()	Read City/State File Next
• z4adrget()	Read ZIP+4 File by Key
• z4adrnxt()	Read ZIP+4 File Next
• z4getzip()	Get a ZIP Code range for a City/St
• z4abort()	Terminate Active Address Inquiry
• z4date()	Get Date of ZIP+4 Database
• z4expire()	Get CD ROM Expiration Information
• z4ver()	Get the Version of the API code
• z4scroll()	Scroll Stack of Address Records

Open the Address Matching System

The z4open() function opens the Address Matching System for application use. This function must be called before attempting to use any of the inquiry functions. During system opening, the Address Matching System allocates memory buffers and file handles for disk I/O. The function returns a code summarizing the results of the open operation.

It is recommended that you use the z4opencfg() function (see page 28) instead of the z4open() function. The z4open() function searches the systems for a configuration file and will use the first one found. This function can cause unexpected operation by using a configuration file that was not expected to be in its search path.

Note: *The Address Matching System does not allocate memory until the z4open() function is called. See Section 2, Installation Procedures (pages 3–20), for information on specific memory requirements for the Address Matching System.*

Syntax

```
#include <zip4.h>
int z4open(void);
```

Input

None

Output

None

Return

0 - The USPS Address Matching System opened successfully

1 - The USPS Address Matching System not resident

2 - The USPS Address Matching System not ready

3 - The USPS Address Matching System has expired

Example

```
#include <stdio.h>
#include <zip4.h>

void main( void)
{
   /*open The USPS Address Matching System */
   if (z4open() == 0)
      printf(“The USPS Address Matching System Open.\n”);
   else
      printf(“Error opening the USPS Address Matching System.\n”);
}
```

Open Address Matching System with Special Parameters

The z4opencfg() function opens the Address Matching System in the same manner as z4open(), but it gives the user more control over the configuration file and LOT processing.

LOT (Line of Travel) is available through the USPS AMS API, but it is turned off by default. To enable LOT processing, you must first call z4opencfg() and set the *lotflag* variable to ‘Y’.

The configuration file can be changed by using the character pointer *fname* to point to the path and filename for a configuration file on your system, or by specifying the paths in the CONIG_PARM character pointers.

Syntax

```
#include <zip4.h>
int z4opencfg(Z4OPEN_PARM*openparm);
```

Input

openparm	pointer to a Z4OPEN_PARM structure.

If a field in the Z4OPEN_PARM is not used, then it must be initialized to zero (see example code).

Output

Z4OPEN_PARM.status will be set to 1, 2 or 9 to indicate which value was used for the configuration file.

Name	Value	Meaning
Z4_FNAME	1	Used the value pointed to by the *fname* character pointer
Z4_CONFIG	2	Used the values pointed to by the CONFIG_PARM structure
Z4_SEARCH	9	Searched for a file named z4config.dat

Return

See z4open().

Example

```
#include <stdio.h>
#include <zip4.h>

void main(void)
{
   Z4OPEN_PARM  openparm;
   int          rtn=0;

   memset(&openparm, 0, sizeof(openparm));

   /*Use the fname character pointer to point to a file on the user’s
system*/
   openparm.fname = “c:\\ams\\special.cfg”;
   /*Turn LOT processing on*/
   openparm.lotflg = ‘Y’

   /*open the USPS Adress Matching System*/
   rtn = z4opencfg(&openparm);

   if(rtn==0)
           printf(“\nThe USPS Address Matching System Opened Success-
fully.”);
   else
           printf(“\nError Opening the USPS Address Matching Sys-
tem.”);

   /*close the USPS Address Matching System*/
   z4close();

   /*Open with the paths embedded in the CONFIG_PARM structure*/
   /*reset variables*/
   memset(&openparm, 0, sizeof(openparm));
   rtn=0;

   /*Setting up paths instead of using the configuration file*/
   openparm.config.address1     =”c:\\amsdata\\”;
   openparm.config.addrindex    =”c:\\amsdata\\”;
   openparm.config.cdrom        =”d:\\”;
   openparm.config.citystate    =”c:\\amsdata\\”;
   openparm.config.crossref     =”c:\\amsdata\\”;
   openparm.config.system       =”c:\\amsdata\\”;

   /*Turn LOT processing on*/
   openparm.lotflag = ‘Y’;

   if(rtn==0)
           printf(“\nThe USPS Address Matching System Opened Success-
fully.”);
   else
           printf(“\nError Opening the USPS Address Matching Sys-
tem.”);

   /*close the USPS Address Matching System*/
   z4close();
}
```

Address Inquiry

The z4adrinq() function commands the Address Matching System to perform an address inquiry using firm name (optional), address, and city/state/ZIP information. Before performing this function, the input address information must be copied into the corresponding input fields outlined below. Note that the City, State, and ZIP fields may be placed either within the parm.ictyi field or copied to the parm.ictyi, parm.stai, and parm.izipc fields, respectively. Following the address inquiry, the parm.retcc field contains a response code summarizing the inquiry results. If an address response was found, standardized address information will be located in the output fields described below.

Syntax

```
#include <zip4.h>
int z4adrinq(ZIP4_PARM *parm);
```

Input

The parm argument must point to a ZIP4_PARM structure. The following fields must be initialized before calling the z4adrinq() function. If a field is not used, it must be initialized to zero.

parm.iadl1	Street Address
parm.iadl2	Firm Name
parm.iad13	Secondary Address
parm.iprurb	Puerto Rican Urbanization Name
parm.ictyi	City or City/State/ZIP
parm.istai	State or empty
parm.izipc	ZIP or empty

Output

parm.retcc	**Response code**
Z4_SINGLE	31 — A single address was found
Z4_DEFAULT	32 — An address was found, but a more specific address could be found with more information
parm.retcc	**Response Code**
Z4_INVADDR	10 — Invalid input address (i.e., contained a dual address)
Z4_INVZIP	11 — Invalid input 5-digit ZIP Code
Z4_INVSTATE	12 — Invalid input state abbreviation code
Z4_INVCITY	13 — Invalid input city name
Z4_NOTFND	21 — No match found using input address
Z4_MULTIPLE	22 — Multiple responses were found and more specific information is required to select a single or default response
parm.foot	**Footnotes**
parm.foot.a = “A”	ZIP Code Corrected

parm.foot.b = “B”	City/State Corrected
parm.foot.c = “C”	Invalid City/State/ZIP
parm.foot.d = “D”	No ZIP+4 Code Assigned
parm.foot.e = “E”	ZIP Code Assigned with a Multiple Response
parm.foot.f = “F”	Address Not Found
parm.fot.g = “G”	All or Part of the Firm Line Used For Address Line
parm.foot.h = “H”	Missing Secondary Number
parm.foot.i = “I”	Insufficient/Incorrect Data
parm.foot.j = “J”	PO Box Dual Address
parm.foot.k = “K”	Non-PO Box Dual Address
parm.foot.l = “L”	Address Component Changed
parm.foot.m = “M”	Street Name Changed
parm foot.n = “N”	Address Standardized
parm.foot.p = “P”	Better Address Exists
parm.foot.s = “S”	Incorrect Secondary Number
parm.foot.t = “T”	Street Address Corrected. This footnote is a composite of the “L,” “M,” and N” footnotes. It is supported for backwards compatibility.
parm.foot.u = “U”	Unofficial Post Office Name
parm.foot.v = “V”	Unverifiable City/State
parm.foot.w = “W”	Small Town Default
parm.foot.x = “X”	Unique ZIP Code Default
parm.foot.z = “Z”	ZIP Move Match

Return Address	**Description**
parm.dadl1	Standardized Output Address
parm.dadl2	Standardized Output Firm Name
parm.dad13	Standardized Secondary Address
parm.dprurb	Standardized Puerto Rican Urbanization Name
parm.dctya	Standardized Output City
parm.dstaa	Standardized Output State
parm.zipc	5-digit ZIP Code
parm.addon	4-digit Add-on Code
parm.cris	4-digit Carrier Route Code
parm.county	3-digit County Code

parm.dpbc	2-digit Delivery Point Barcode and 1-digit Check Digit
parm.mpnum	Matched Primary Number
parm.msnum	Matched Secondary Number
parm.auto_zone_ind	Carrier Route Rate Sort Indicator (Y or N)
parm.lot_num	Line of Trave (LOT) number
parm.lot_code	LOT Ascending/Descending Flag (A/D)

Parsed Input	**Description**
ppnum	Primary Number
psnum	Secondary Number
prote	Rural Route Number
punit	Secondary Number Unit
ppre1	First or Left Pre-direction
ppre2	Second or Right Pre-direction
psuf1	First or Left Suffix
psuf2	Second or Right Suffix
ppst1	First or Left Post-direction
ppst2	Second or Right Post-direction
ppnam	Primary Name

Return

0 - The USPS Address Matching System resident

1 - The USPS Address Matching System issued a system error

2 - The USPS Address Matching System not ready

3 - CD-ROM has expired

Additional Information About Z4ADRINQ()

If parm.retcc is Z4_INVADDR, Z4_INVZIP, Z4_INVSTATE, Z4_INVCITY, Z4_NOTFND, or Z4_MULTIPLE, then the return address fields will contain the input address. If the input address is unambiguously a rural route, highway contract, PO box, or general delivery address, then the return fields will contain the normalized version of the input address.

If parm.retcc is Z4_MULTIPLE, then parm.foot, parm.respon, and parm.stack are also returned by the system. The parm.zipc and/or parm.cris fields may contain data if all records in the stack have the same ZIP Code and/or carrier route ID.

If parm.retcc is Z4_SINGLE or Z4_DEFAULT, then all fields in the returned data section are returned by the Address Matching System. The first record in the parm.stack structure will contain the ZIP+4 record to which the system matched. This record may be used to access the individual fields from the matched record, such as primary name, suffix, post-directional, etc.

Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <zip4.h>
ZIP4_PARM parm;
void main(void)
{
   /* open The USPS Address Matching System */
   if (z4open() != 0)
   {
      printf(“The USPS Address Matching System not resident.\n”);
      exit(5);
   }

   /* load input address parameters */
   memset(&parm, 0, sizeof(parm));
   strcpy(parm.iadl1,  ”323 S 152ND ST”);
   strcpy(parm.iad13,  ”STE 200”);
   strcpy(parm.iadl2,  ”ACME TOOL AND DIE”);
   strcpy(parm.iprurb, ””);
   strcpy(parm.ictyi,  ”OMAHA, NE 68154");
   /* request address inquiry */
   z4adrinq(&parm);
   /* if a response found (either single or default) */
   if(parm.retcc==Z4_SINGLE || parm.retcc==Z4_DEFAULT)
   {
      printf(“Found response.\n”);
      printf(“Name:     %s\n”, parm.dadl2);
      printf(“S Addr:   %s\n”,parm.dad13);
      printf(“Addr:     %s\n”, parm.dadl1);
      printf(“PRUrb:    %s\n”, parm.dprurb);
      printf(“City:     %s\n”, parm.dctya);
      printf(“ST:       %s\n”, parm.dstaa);
      printf(“ZIP:      %s\n”, parm.zipc);
      printf(“Addon:    %s\n”, parm.addon);
      printf(“DPBC:     %s\n”, parm.dpbc);
      printf(“Pre Dir:  %s\n”, parm.stack[0].pre_dir);
      printf(“Str Name: %s\n”, parm.stack[0].str_name);
      printf(“Suffix:   %s\n”, parm.stack[0].suffix);
      printf(“Post Dir: %s\n”, parm.stack[0].post_dir);
      printf(“Lacs Ind: %c\n”, parm.stack[0].lacs_status);
   }

   /* close The USPS Address Matching System */
   z4close();
   exit(0)
}
```

Address Sort Key

The z4adrkey() function creates a sort key for an address. This function can be used in batch processes to sort an input file in the order that addresses are contained on the Address Matching System data files. However, the function does not sort your file; it produces a key field to assist your software in sortation. Sorting an input file usually produces a dramatic increase in processing throughput.

Syntax

```
#include <zip4.h>
int z4adrkey(ZIP4_PARM *parm);
```

Input

The parm argument must point to a ZIP4_PARM structure. The following fields must be initialized before calling the z4adrkey() function.

parm.iadl1	Street Address
parm.iadl2	Firm Name
parm.iprurb	Puerto Rican Urbanization Name
parm.ictyi	City or City/ State/ ZIP
parm.istai	State or empty
parm.izipc	ZIP or empty

Output

parm.adrkey	Address Sort Key

Note: *Do not make any assumptions as to the contents or length of the address sort key: both are subject to change at any time. The key contains binary data and should be used in its entirety for the sort process.*

Return

0 - The USPS Address Matching System resident

1 - The USPS Address Matching System issued a system error

2 - The USPS Address Matching System not ready

Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <zip4.h>

ZIP4_PARM parm;

void main(void)
{
   int i;
   /* open The USPS Address Matching System */
   if (z4open() != 0)
   {
      printf(“The USPS Address Matching System not resident.\n”);
      exit(5);
   }


   /* load input address parameters */
   memset(&parm, 0, sizeof(parm));

   strcpy(parm.iadl1, ”323 S 152ND ST”);
   strcpy(parm.iadl2, ”ACME TOOL AND DIE”);
   strcpy(parm.ictyi, ”OMAHA, NE 68154");

   /* request address sort key */
   z4adrkey(&parm);

   /* print the address sort key in hex */
   for(i=0; i<sizeof(parm.adrkey); i++)
      printf(“%02X“, parm.adrkey[i]);
   printf(“\n”);

   /* close The USPS Address Matching System */
   z4close();
   exit(0);
}
```

9-digit Inquiry

The z4xrfinq() (9-digit Inquiry) function commands the Address Matching System to perform an address inquiry using an input 9-digit ZIP Code. Before using this function, the input 9-digit ZIP Code must be copied into the parm.iad11 field outlined below. Following the 9-digit inquiry, the parm.retcc field displays a return code summarizing the result of the inquiry. If an address response was found, standardized address information can be found in the output fields described in the Address Inquiry function description (see page 24).

Syntax

```
#include <zip4.h>
int z4xrfinq(ZIP4_PARM *parm);
```

Input

The parm argument must point to a ZIP4_PARM structure. The following field must be initialized before calling the z4xrfinq() function:

parm.iadl1	9-digit ZIP Code.

Note: *Return Code 22 denotes multiple responses. The address fields contain the first of a stack of ten possible responses (or matches). The first address in the output fields should not be used as a mailing address because it is not an exact match.*

Output

parm.retcc	**Response code**
Z4_SINGLE	A single address was found
Z4_DEFAULT	A default address was found, but more specific addresses exist
Z4_NOTFND	No match found; considered a not found address
Z4_MULTIPLE	Multiple responses were found

Refer to the Address Inquiry function description for other output fields (see page 29).

Return

0 - The USPS Address Matching System resident

1 - The USPS Address Matching System issued a system error

2 - The USPS Address Matching System not ready

3 - The USPS Address Matching System has expired

Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <zip4.h>
```

ZIP4_PARM parm;

```
void main(void)
{
   /* check for The USPS Address Matching System residence */
   if(z4open() != 0)
   {
      printf(“The USPS Address Matching System not resident.\n”);
      exit(5);
   }


   /* load input 9-digit ZIP parameter */
   memset(&parm, 0, sizeof(parm));
   strcpy(parm.iadl1, ”681642815");

   /* request address inquiry */
   z4xrfinq(&parm);

   /* if a response found (either single or default) */
   if(parm.retcc == Z4_SINGLE || parm.retcc == Z4_DEFAULT)
   {
      printf(“Found response.\n”);
      printf(“Name:  %s\n”, parm.dadl2);
      printf(“Addr:  %s\n”, parm.dadl1);
      printf(“PRUrb: %s\n”, parm.dprurb);
      printf(“City:  %s\n”, parm.dctya);
      printf(“ST:    %s\n”, parm.dstaa);
      printf(“ZIP:   %s\n”, parm.zipc);
      printf(“Addon: %s\n”, parm.addon);
      printf(“DPBC:  %s\n”, parm.dpbc);
   }

   /* close The USPS Address Matching System */
   z4close();
   exit(0);
}
```

Address Standardization

The z4adrstd() (Address Standardization) function instructs the Address Matching System to standardize an address. This function can be used when a Z4_MULTIPLE response is returned from the z4adrinq() function. Use this function to standardize an address from the stack, but use it with caution. The index parameter is relative to zero and must be in increments of ten for each Z4scroll() function called. Therefore, the index will have a value between zero and parm.respn minus one. Do not use the offset into the current stack of ten records.

When this function is called, the record corresponding to the index value is moved to the first position on the stack (offset zero). If components from the ADDR_REC structure are needed for the current record that was processed through z4adrstd(), they may be retrieved from the first stack record. Do not use the modulus 10 of the index (index % 10) to retrieve the ADDR_REC components from the stack.

Note: *This function should only be used when an operator is reviewing the multiple responses returned and selecting the record to be standardized. Using this function in an unattended (batch) mode will result in inaccurate matches and possible failure to CASS certify.*

Syntax

```
#include <zip4.h>
int z4adrstd(ZIP4_PARM *parm, int index)
```

Input

parm	Unmodified parameter list from previous call to z4adrinq().
index	Index of stack record to standardize address (refer to the description above). This must be less than parm.respn.

Output

parm.dadl1	Standardized Street Address
parm.dadl2	Standardized Firm Name
parm.dprurb	Standardized Puerto Rican Urbanization Name
parm.dlast	Standardized City/State/ZIP

Return

0 - Success

1 - Failure (i.e., invalid index parameter)

2 - The USPS Address Matching System not ready

Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <zip4.h>
ZIP4_PARM parm;
```

```
void main(void)
{
   /* check for The USPS Address Matching System residence */
   if(z4open() != 0)
   {
      printf(“The USPS Address Matching System not resident.\n”);
      exit(5);
   }

   /* load input address parameters */
   memset(&parm, 0, sizeof(parm));
   strcpy(parm.iadl1, ”DODGE ST”);
   strcpy(parm.iadl2, ””);
   strcpy(parm.iprurb, ””);
   strcpy(parm.ictyi, ”OMAHA NE”);

   /* request address inquiry */
   z4adrinq(&parm);

   /* standardize second address */
   z4adrstd(&parm, 1);

   /* display address */
   printf(“Found response.\n”);
   printf(“Name:  %s\n”, parm.dadl2);
   printf(“Addr:  %s\n”, parm.dadl1);
   printf(“PRUrb: %s\n”, parm.dprurb);
   printf(“City:  %s\n”, parm.dctya);
   printf(“ST:    %s\n”, parm.dstaa);
   printf(“ZIP:   %s\n”, parm.zipc);
   printf(“Addon: %s\n”, parm.addon);
   printf(“DPBC:  %s\n”, parm.dpbc);
   /* close The USPS Address Matching System */
   z4close();
   exit(0);
}
```

Close the Address Matching System

The z4close() function closes the Address Matching System and is called when address inquiries have been completed and the interface is no longer needed. During execution of this function, memory buffers and file handles allocated during the z4open() function are de-allocated and closed.

Syntax

```
#include <zip4.h>
int z4close(void);
```

Input

None

Output

None

Return

0 - The USPS Address Matching System closed

1 - The USPS Address Matching System not resident

2 - The USPS Address Matching System not ready

Example

```
#include <stdio.h>
#include <zip4.h>

void main(void)
{
   /* close The USPS Address Matching System */
   if(z4close() == 0)
      printf(“The USPS Address Matching System closed.\n”);
   else
      printf(“Error closing the USPS Address Matching System.\n”);
}
```

Read City/State File By Key

The z4ctyget() (Read City/State File By Key) function initiates a read of the City/State File. A specific ZIP Code can be selected as a starting point in a read of the City/State File. To read subsequent records, the Read City/State File Next function is used. For documentation on the City/State File, please refer to the *Address Information System Products Technical Guide*, which is available from the USPS National Customer Support Center's Customer Service Department by calling (800) 238-3150.

Syntax

```
#include <zip4.h>
int z4ctyget(CITY_REC *cityrec, char *zipcode);
```

Input

The CITYREC argument must point to a CITY_REC structure. The contents of the structure will be altered to contain the first city for the requested ZIP Code. The ZIP Code argument must point to a valid 5-digit ZIP Code or "00000."

Output

None

Return

0 - Success

1 - Failure

2 - The USPS Address Matching System not ready

Read City/State File Next

The z4ctynext() (Read City/State File Next) function reads subsequent records of the City/State File. It can only be used after the z4ctynext() function has been called.

Syntax

```
#include <zip4.h>
int z4ctynxt(CITY_REC *cityrec);
```

Input

The CITYREC argument must point to a CITY_REC structure. The contents of the structure will be altered to contain the next city.

Output

None

Return

0 - Success

1 - Failure

2 - The USPS Address Matching System not ready

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <zip4.h>

CITY_REC city;

void main(void)
{
   int i;
   /* open The USPS Address Matching System */
   if(z4open() != 0)
   {
      printf(“The USPS Address Matching System not resident.\n”);
      exit(5);
   }
   /* read first city */
   z4ctyget(&city, “00000”);
   /* read 10 more cities */
   for(i=0; i<10 && z4ctynext(&city) == 0; ++i)
   {
      printf(“%s %-28.28s %s %s\n”, city.zip_code, city.city_name,
            city.state_abbrev, city.finance);
   }
   /* close The USPS Address Matching System */
   z4close();
   exit(0);
}
```

Read ZIP+4 File By Key

The z4adrget() (Read ZIP+4 File by Key) function is used to read the ZIP+4 File. For documentation of the ZIP+4 File, please refer to the *Address Information Products Technical Guide*, which is available from the USPS National Customer Support Center's Customer Service Department by calling (800) 238-3150.

A specific postal finance number can be selected as a starting point in a read of the ZIP+4 File. To read subsequent records, the z4adrget() function is used.

Syntax

```
#include <zip4.h>
int z4adrget(ADDR_REC *addrrec, char *finance);
```

Input

The ADDRREC argument must point to an ADDR_REC structure. The contents of the structure will be altered to contain the first address for the requested postal finance number. The finance argument must contain a valid postal finance number or "000000."

Output

None

Return

0 - Success

1 - Failure

2 - The USPS Address Matching System not ready

Read ZIP+4 File Next

The z4adrnxt() (Read ZIP+4 File Next) function reads subsequent records of the ZIP+4 File. It can only be used after the z4adrnxt() function has been called.

Syntax

```
#include <zip4.h>
int z4adrnxt(ADDR_REC *addrrec);
```

Input

The ADDRREC argument must point to a ADDR_REC structure. The contents of the structure will be altered to contain the next address.

Output

None

Return

0 - Success

1 - Failure

2 - The USPS Address Matching System not ready

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <zip4.h>

CITY_REC city;
ADDR_REC addr;

void main(void)
{
   /* open The USPS Address Matching System */
   if (z4open() != 0)
   {
      printf(“The USPS Address Matching System not resident.\n”);
      exit(5);
   }

   /* read a city */
   z4ctyget(&city, “00000”);

   /* read first address record for this city */
   z4adrget(&addr, city.finance);


   /* read remaining addrs for this finance number */
   while(z4adrnxt(&addr) == 0)
   {
      /* check if finance number has changed */
      if (memcmp(addr.finance, city.finance, 6) != 0)
            break;

      /* Code to process the current address record. */
   }

   /* close The USPS Address Matching System */
   z4close();
   exit(0);
}
```

Read ZIP+4 File Next

The z4adrnxt() (Read ZIP+4 File Next) function reads subsequent records of the ZIP+4 File. It can only be used after the z4adrnxt() function has been called.

Syntax

```
#include <zip4.h>
int z4adrnxt(ADDR_REC *addrrec);
```

Input

The ADDRREC argument must point to a ADDR_REC structure. The contents of the structure will be altered to contain the next address.

Output

None

Return

0 - Success

1 - Failure

2 - The USPS Address Matching System not ready

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <zip4.h>

CITY_REC city;
ADDR_REC addr;

void main(void)
{
   /* open The USPS Address Matching System */
   if (z4open() != 0)
   {
      printf("The USPS Address Matching System not resident.\n");
      exit(5);
   }

   /* read a city */
   z4ctyget(&city, "00000");

   /* read first address record for this city */
   z4adrget(&addr, city.finance);
```

```
/* read remaining addrs for this finance number */
while(z4adrnxt(&addr) == 0)
{
   /* check if finance number has changed */
   if (memcmp(addr.finance, city.finance, 6) != 0)
         break;

   /* Code to process the current address record. */
}

/* close The USPS Address Matching System */
z4close();
exit(0);
```

Get ZIP Codes from a City/State

The z4getzip() (Get ZIP Codes) from a City/State function retrieves a range of ZIP Codes for a city or state and returns the valid high and the low values for the input city/state. The standardized form of the input city/state as well as the finance number are also returned.

***Note**: All ZIP Codes within the range are not necessarily valid.*

Syntax

```
#include <zip4.h>
int z4getzip(GET_ZIPCODE_STRUCT *parm);
```

Input

The GETZIP_STRUCT argument must point to a GET_ZIPCODE_STRUCT structure. The contents of the structure will be altered to contain the ZIP Code range for the input city/state.

parm.input_cityst	Input city/state to lookup

Output

parm.output_cityst	Standardized city/state
parm.low_zipcode	Low ZIP Code value
parm.high_zipcode	High ZIP Code value
parm.finance_num	Finance number

Return

0 - Success

1 - Failure

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <zip4.h>

GET_ZIPCODE_STRUCT parm;

void main(void)
{
   int result;

   /* open The USPS Address Matching System */
   if (z4open() != 0)
   {
      printf("The USPS Address Matching System not resident.\n");
      exit(5);
   }
   /* read a city */
   strcpy(parm.input_cityst, “MEMPHIS TN”);
   result=z4getzip(&parm);

   /* Display the ZIP codes found */
   if(result == 1)
   {
      printf(“CITY FOUND: %s\n”, parm.output_cityst);
      printf(“LOW ZIP:    %s\n”, parm.low_zipcode);
      printf(“HIGH ZIP:   %s\n”, parm.high_zipcode);
      printf(“FINANCE:    %s\n”, parm.finance_num);
   }

   /* close The USPS Address Matching System */
   z4close();
   exit(0);
}
```

Terminate Active Address Inquiry

The z4abort() (Terminate Active Address Inquiry) function terminates an active address inquiry and is useful in real-time applications where each inquiry must be completed within a specified period of time. This function would normally be called from within a timer interrupt handler (i.e., INT 08H or INT 1CH for DOS). The zadrinq() call in progress is terminated by the function call. The zadrinq() function has a Return Code 21 (Not Found).

Syntax

```
#include <zip4.h>
int z4abort(void);
```

Input

None

Output

None

Return

None

Get Date of ZIP+4 Database

The z4date() (Get Date of ZIP+4 Database) function returns the date of the ZIP+4 database and prints "DATE OF ZIP+4 DATABASE USED" for USPS Form 3553. The date is returned as an 8-byte character string in the "YYYYMMDD" format.

Note: *The date was previously returned in a 6-byte string but, effective with the CASS 1998 release of the API, has been expanded to eight bytes to accommodate the year 2000.*

Syntax

```
#include <zip4.h>
int z4date(char *date);
```

Input

Address of field to return the date of the ZIP+4 database. This field must be at least nine bytes in length.

Output

The date of the ZIP+4 database. This field must be at least nine bytes in length.

Return

0 - Success

1 - Failure

2 - The USPS Address Matching System not ready

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <zip4.h>

char date[9];

void main(void)
{
   int i;
   /* open The USPS Address Matching System */
   if (z4open() != 0)
   {
      printf("The USPS Address Matching System not resident.\n");
      exit(5);
   }

   /* get release date */
   z4date(date);
   printf("Release date: %s\n", date);

   /* close The USPS Address Matching System */
   z4close();
   exit(0);
}
```

Get CD-ROM Expiration Information

The z4expire() (Get CD-ROM Expiration Information) function instructs the Address Matching System to return the number of days until the CD-ROM expires. Because the function can be used periodically to check the number of days remaining until CD-ROM expiration, it is strongly recommended that you integrate this function into your software.

Syntax

```
#include <zip4.h>
int z4expire(void);
```

Input

None

Output

None

Return

-1 - CD-ROM has expired. Otherwise, the function returns the number of days until CD-ROM expiration.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <zip4.h>

void main(void)
{
   int days;
   /* open The USPS Address Matching System */
   if (z4open() != 0)
   {
      printf(“The USPS Address Matching System not resident.\n”);
      exit(1);
   }

   /* get number of days until CD-ROM expiration */
   days = z4expire();
   if (days == -1)
      printf(“CD-ROM has already expired.\n”);
   else
      printf(“%d days until CD-ROM expiration.\n”, days);

   /* close The USPS Address Matching System */
   z4close();
   exit(0);
}
```

Get API Code Version

The z4ver() (Get API Code Version) function commands the program to retrieve the version string of the API code. This string is in compliance with the CASS requirements for address-matching software version information and may be used when generating a PS Form 3553 for mailing discounts.

Syntax

```
#include <zip4.h>
int z4ver(char *str);
```

Input

none

Output

str	pointer to data butter to receive the string

Return

0 - Success

Example

```
#include <stdio.h>
#include <zip4.h>

void main(void)
{
    char version[32];
   /* get the Address Matching System version */
   z4ver(version) ;

   printf(“The Address Matching System version is %s\n”, version) ;
   exit (0);
}
```

Scroll Stack of Address Records

The z4scroll() (Scroll Stack of Address Records) function commands the Address Matching System to access additional stacks of ten address records each. The function is related to the z4adrinq() and z4xrfinq() functions, which return up to ten records when the Z4_MULTIPLE or Z4_DEFAULT return codes are set. When the parm.respn field contains a number greater than ten, your program can use this function to obtain additional stacks of ten address records (up to the number of records specified in the parm.respn return field). This function may only be called immediately after a call to the z4adrinq() or z4xrfinq() functions.

Syntax

```
#include <zip4.h>
int z4scroll(parm);
ZIP4_PARM *parm;
```

Input

The parm argument must point to a ZIP4_PARM structure. This structure should not be modified after the call to z4adrinq().

Output

The parm.stack field will be updated to contain the next ten records (fewer records may be returned if less than ten records remain).

Return

0 - Success

1 - The USPS Address Matching System not installed

2 - The USPS Address Matching System not open

3 - Stack access not allowed

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <zip4.h>

ZIP4_PARM  parm;

void main(void)
{
   int i;

   /* open The USPS Address Matching System */
   if (z4open())
   {
      printf("Error opening The USPS Address Matching System\n");
      exit(1);
   }
   /* create parameter list and call The USPS Address Matching System
*/
   memset(&parm, 0, sizeof(parm));
   strcpy(parm.iadl1, “350 5TH AVE”);
   strcpy(parm.ictyi, “NEW YORK NY”);
   z4adrinq(&parm);

   /* process all addresses returned by The USPS Address Matching System
*/
   for(i=0; i<parm.respn; i++)
   {

   /* check if stack needs to be refreshed */
   if (i != 0 && (i% 10) == 0)
      {
            if(z4scroll(&parm))
                  break;
   }

   /* examine each address returned by The USPS Address Matching System
*/
   ...
   }

   /* close The USPS Address Matching System */
   z4close();
   exit(0);
}
```

Section 4: Footnote Flags

A **ZIP CODE CORRECTED**

The address was found to have a different 5-digit ZIP Code than given in the submitted list. The correct ZIP Code is shown in the output address.

B **CITY / STATE SPELLING CORRECTED**

The spelling of the city name and/or state abbreviation in the submitted address was found to be different than the standard spelling. The standard spelling of the city name and state abbreviation are shown in the output address.

J **PO BOX DUAL ADDRESS**

The input address contained both a PO BOX address and a non-PO BOX address. A match was made using the PO BOX address. For example, if the input address were 123 MAIN ST PO BOX 99, the output address would be PO BOX 99.

K **NON-PO BOX DUAL ADDRESS**

The input address contained both a PO BOX address and a non-PO BOX address. A match was made using the non-PO BOX address. For example, if the input address were 123 MAIN ST PO BOX 99, the

output address would be 123 MAIN ST.

L **ADDRESS COMPONENT CHANGED**

An address component (i.e., directional or suffix only) was added, changed, or deleted in order to achieve a match.

M **STREET NAME CHANGED**

The spelling of the street name was changed in order to achieve a match.

N **ADDRESS STANDARDIZED**

The delivery address was standardized. For example, if STREET was in the delivery address, the system will return ST as its standard spelling.

U **UNOFFICIAL POST OFFICE NAME**

The city or post office name in the submitted address is not recognized by the United States Postal Service as an official last line name (preferred city name), and is not acceptable as an alternate name. This does denote an error and the preferred city name will be provided as output.

C **INVALID CITY / STATE / ZIP**

The ZIP Code in the submitted address could not be found because neither a valid city, state, nor valid 5-digit ZIP Code was present. It is also recommended that the requestor check the submitted address for accuracy.

D **NO ZIP+4 ASSIGNED**

This is a record listed by the United States Postal Service on the national ZIP+4 file as a non-deliverable location. It is recommended that the requestor verify the accuracy of the submitted address.

F **ADDRESS COULD NOT BE FOUND IN THE NATIONAL DIRECTORY FILE DATABASE**

The address, exactly as submitted, could not be found in the city, state, or ZIP Code provided. It is also recommended that the requestor check the submitted address for accuracy. For example, the street address line may be abbreviated excessively and may not be fully recognizable.

H **MISSING SECONDARY NUMBER**

ZIP+4 information indicates this address is a building. The address as submitted does not contain an apartment/suite number. It is recommended that the requestor check the submitted address and add the missing apartment or suite number to ensure the correct Delivery Point Barcode (DPBC).

I **INSUFFICIENT / INCORRECT ADDRESS DATA**

More than one ZIP+4 Code was found to satisfy the address as submitted. The submitted address did not contain sufficiently complete or correct data to determine a single ZIP+4 Code. It is recommended that the requestor check the address for accuracy and completeness. For example, firm name, or institution name, doctor's name, suite number, apartment number, box number, floor number, etc. may be missing or incorrect. Also pre-directional or post-directional indicators (North = N, South = S, East = E, West = W, etc.) and/or street suffixes (Street = ST, Avenue = AVE, Road = RD, Circle = CIR, etc.) may be missing or incorrect.

P BETTER ADDRESS EXISTS

The delivery address is matchable, but is known by another (preferred) name. For example, in New York, NY, AVENUE OF THE AMERICAS is also known as 6TH AVE. An inquiry using a delivery address of 55 AVE OF THE AMERICAS would be flagged with a Footnote Flag P.

S INCORRECT SECONDARY ADDRESS

The secondary information (i.e., floor, suite, apartment, or box number) does notmatch that on the national ZIP+4 file. This secondary information, althoughpresent on the input address, was not valid in the range found on the national ZIP+4 file.

T MULTIPLE RESPONSE DUE TO MAGNET STREET SYNDROME

The search resulted in a single response; however, the record matched was flagged as having magnet street syndrome and the input street name components (pre-directional, primary street name, post-directional, and suffix) did not exactly match those of the record. A "magnet street" is one having a primary street name that is also a suffix or a directional word, having either a post-directional or a suffix (i.e., 2200 PARK MEMPHIS TN logically matches to a ZIP+4 record 2200-2258 PARK AVE MEMPHIS TN 38114-6610), but the input address lacks the suffix 'AVE' which is present on the ZIP+4 record. The primary street name 'PARK' is a suffix word. The record has either a suffix or a post-directional present. Therefore, in accordance with CASS requirements, a ZIP+4 Code must not be returned. The multiple response return code is given since a “no match” would prevent access to the best candidate.

V UNVERIFIABLE CITY / STATE

The city and state in the submitted address could not be verified as corresponding to the given 5-digit ZIP Code. This comment does not necessarily denote an error; however, it is recommended that the requestor check the city and state in the submitted address for accuracy.

W INVALID DELIVERY ADDRESS

The input address record contains a delivery address other than a PO BOX, General Delivery, or Postmaster 5-digit ZIP Code that is identified as a “small town default”. The United States Postal Service does not provide street delivery for this ZIP Code. The United States Postal Service requires use of PO BOX, General Delivery, or Postmaster for delivery within this ZIP Code.

Z MATCH MADE USING THE ZIPMOVE PRODUCT DATA

The ZIPMOVE product shows which ZIP+4 records have moved from one ZIP Code to another. If an input address matches to a ZIP+4 record which the ZIPMOVE product indicates as having moved, the search is performed again in the new ZIP Code.

Section 4: Record Types

F FIRM

This is a match to a Firm Record, which is the finest level of match available for an address.

G GENERAL DELIVERY

This is a match to a General Delivery record.

H BUILDING / APARTMENT

This is a match to a Building or Apartment record.

R HIGHWAY CONTRACT

This is a match to a Highway Contract record, which may have associated Box Number ranges.

R RURAL ROUTE

This is a match to a Rural Route record, which may have associated Box Number ranges.

P POST OFFICE BOX

This is a match to a Post Office Box.

S STREET RECORD

This is a match to a Street record containing a valid primary number range.

Section 4: Return Codes

10 INVALID DUAL ADDRESS

Information presented could not be processed in current format. Corrective action is needed. Be sure that the address line components are correct. For example, the input address line may contain more than one delivery address.

11 INVALID CITY/ST/ZIP

The ZIP Code in the submitted address could not be found because neither a valid city, state, nor valid 5-digit ZIP Code was present. Corrective action is needed. It is also recommended that the requestor check the submitted address for accuracy.

12 INVALID STATE

The state in the submitted address is invalid. Corrective action is needed. It is also recommended that the requestor check the submitted address for accuracy.

13 INVALID CITY

The city in the submitted address is invalid. Corrective action is needed. It is also recommended that the requestor check the submitted address for accuracy.

21 NOT FOUND

The address, exactly as submitted, could not be found in the national ZIP+4 file. It is recommended that the requestor check the submitted address for accuracy. For example, the street address line may be abbreviated excessively and may not be fully recognizable.

22 MULTIPLE RESPONSE

More than one ZIP+4 Code was found to satisfy the address submitted. The submitted address did not contain sufficiently complete or correct data to determine a single ZIP+4 Code. It is recommended that the requestor check the address for accuracy and completeness. Address elements may be missing.

31 EXACT MATCH

Single response based on input information. No corrective action is needed since an exact match was found in the national ZIP+4 file.

32 DEFAULT MATCH

A match was made to a default record in the national ZIP+4 file. A more specific match may be available if a secondary number (i.e., apartment, suite, etc.) exists.

Appendix A - Sample Program

```
/*************************************************************************/
/*   Program            - Sample                                         */
/*   Purpose            - Demonstrate the basic use of the USPS AMS API */
/*   Date Modified      - 14 February 2001                               */
/*   Date Original      - 22 December 1994                               */
/*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <string.h>
#include <zip4.h>

static ZIP$_PARM parm;

/*local functions*/

void main(void);
static int display(void);

void main(void)
{
      char  version[20] ={0},
            date [9]    ={0},
            ddate[11]   ={0};

      int   rtncode     =0,
            days        =0;

      /*LOT processing is off by default. z4opencfg must be called*/
      /*If LOT processing is needed.                               */
      Z4OPEN_PARM openparm;

      /*It is always good to start out with a clean variable*/
      memset(&openparm, 0, sizeof(openparm));

      /*Z4OPEN_PARM allows more control over opening AMS    */
      /*if both the fname pointer and the CONFIG struct are */
      /*blank, then AMS will search for a z4config.dat file */

      openparm.lotflg = ‘y’;  /*turn LOT processing on      */

      /*z4ver can be called without opening the engine first*/
      z4ver(version);

      /*Open USPS AMS API     */
      if ((rtncode = z4opencfg(&openparm))
      {
       printf(“\nError opening USPS AMS API - Version %s\n\n”, version);

       puts(“Common Problems”);
       puts(“---------------”);
     switch(rtncode)
     {
        case 1;
            puts(“1. Is z4config.dat pointing to the correct directories?”);
            puts(“2. Are the database files from a different CASS version?”);
            puts(“3. Are the database files all from the same AMS CD?”);
            break;
        case 2;
            puts(“1. Is z4config.dat in the same directory as sample.exe?”);
            puts(“2. Are the files located where z4config.dat points?”);
            puts(“3. Are the database files older than 105 days?”);
            puts(“4. Is it around August 1st? (code expires every year)”);
            puts(“5. Security file may be corrupt. (z4cxlog.dat)”);
      }
      printf(“\n\nz4open() return code = %i\n”, rtncode);
      exit(1);
   }

   /*retrieving the date of the USPS AMS API database*/
   if(z4date(date))
      printf(“\nError retrieving the date of the database”);
   else
   {
      strncopy    (ddate, &date[4], 2);
      strcat      (ddate, “/”);
      strncat     (ddate, &date[6], 2);
      strcat      (ddate, “/”);
      strncat     (ddate, date, 4);
   }

   printf(“\r\n------------------------------------”);
   printf(“\r\n|         USPS AMS API             |”);
   printf(“\r\n|      Sample Application          |”);
   printf(“\r\n|                                  |”);
   printf(“\r\n|   Version        Database Date   |”);
   printf(“\r\n|   -------        -------------   |”);
   printf(“\r\n|    %-10.10s      %10.10s       |           ”,version, ddate);
   printf(“\r\n|                                  |”);
   printf(“\r\n------------------------------------”);

   /*retrieving the number of days until the USPS AMS API database expires*/
   if((days = z4expire()) == -1)
      printf(“\r\n| The USPS AMS API Database has expired         |”);
   else
      printf(“\r\n| The USPS AMS API Database will expire in %i day%c   |”,
days, ((days > 1) ? ‘s’ : ‘ ‘));

   printf(“\r\n-------------------------------------------------\r\n”);

   for (;;)
   {
      printf(“\r\n                 Address Lookup                     “);
      printf(“\r\n----------------------------------------------------\r\n”);
      /* clear structure           */
      memset(&parm, 0x00, sizeof(parm));

      /* get firm name             */
      printf(“\r\nFirm Name           : ”);
      gets(parm.iad12);

      /* get delivery address line 3      */
      printf(“Delivery Address Xtra: ”);
      gets(parm.iad13);

      /* get delivery address line        */
      printf(“Delivery Address     : ”);
      gets(parm.iad11);

      /* get City, State, and ZIP Code    */
      printf(“Last Line            : ”);
      gets(parm.ictyi);

      /* get Urbanization                 */
      printf(“Urbanization          : ”);
      gets(parm.iprurb);

      /* Call UPS AMS API with an address */
      if (display() == 1)
            break;
   }

   /* Close USPS AMS API */
   if(z4close())
      printf(“\r\nError closing USPS AMS API\r\n”);
   else
      printf(“\r\nUSPS AMS API is closed\r\n”);
   exit(0);
}

static int display(void)
{
   int      ch,
            rtncode;

   printf(“\r\n---------------------------------------”);
   printf(“\r\n---------------------------------------”);
   switch(parm.retcc)
   {
      case 32:
        printf(“\r\n    DEFAULT MATCH     Return Code = %i”, parm.retcc);
        break;
      case 31:
        printf(“\r\n    EXACT MATCH       Return Code = %i”, parm.retcc);
        break;
      case 22:
        printf(“\r\n    MULTIPLE RESPONSE Return Code = %i”, parm.retcc);
        break;
      case 21:
        printf("\r\n    ADDRESS NOT FOUND Return Code = %i", parm.retcc);
        break;
      case 13:
        printf("\r\n    INVALID CITY      Return Code = %i", parm.retcc);
        break;
      case 12:
        printf("\r\n    INVALID STATE     Return Code = %i", parm.retcc);
        break;
      case 11:
        printf("\r\n    INVALID ZIP CODE  Return Code = %i", parm.retcc);
        break;
      case 10:
        printf("\r\n    INVALID ADDRESS   Return Code = %i", parm.retcc);
        break;
      default:
        printf("\r\n    ERROR             Return Code = %1", parm.retcc);
   }

   printf("\r\n----------------------------------------");
   printf("\r\n");

   if (parm.retcc >= 31)   /*exact or default match */
   {
      printf("\r\nrecord Type : %c        Finance : %s\r\n", parm.stack-
>rec_type, parm.stack->finance);
      printf("\r\nFirm Name               : %s", parm.dad12);
      printf("\r\nDelivery Address Xtra   : %s", parm.dad13);
      printf("\r\nDevlivery Address       : %s", parm.dad11);

      printf("\r\nLast Line               :%s %s %s-%s  CR %s\r\n",
            parm.dctya, parm.dstaa, parm.zipc, parm.addon, parm.cris);

      printf("\nLOT Number: %s%c", parm.lot_num, parm.lot_code);

      /* Automatic zone indicator flag. If yes, the postal facility     */
      /* does give discounts for carrier route presorted mail.          */
      /* 12 FEBRUARY 2001 - This changed to 4 values last year A/B/C/D  */

      printf("\r\n");
      printf("\r\nAuto Zone Indicator Value = %c\r\n", parm.auto_zone_ind);

      switch(parm.auto_zone_ind)
      {
        case 'A':
            printf("\tCarririer Route Sortation Rates Apply\r\n\tMerging Per-
mitted");
            break;
        case 'B':
            printf("\tCarrier Route Sortation Rates Apply\r\n\tMerging Not
Permitted");
            break;
        case 'C':
            printf("\tCarrier Route Sortation Rates Do Not Apply\r\n\tMerging
Permitted”);
            break;
        case ‘D’:
            printf(“tCarrier Route Sortation Rates Do Not Apply\r\n\tMerging
Not Permitted”);
        default:
            printf(“\tUnknown Carrier Route Sortation Value\r\n\tKnown Values
are A/B/C/D”);
      }

   }
   else if (parm.retcc == 22  /* multiple response */
   {
      printf(“Multiple Response items from stack. Showing %i of %i\r\n”,
(parm.respn < 10) ? parm.respn : 10, parm.respn);

      /* display the 1st 10 items return on the stack */
      for (ch=0; ch<10 && ch<parm.respn; ch++)
        printf(“(%s-%s %c %-2.2s %-28.28s %-4.4s %-2.2s %-04.4s-%04.4s\r\n”;
            parm.stack[ch].prim_low,
            parm.stack[ch].prim_high,
            parm.stack[ch].prim_code,
            parm.stack[ch].pre_dir,
            parm.stack[ch].str_name,
            parm.stack[ch].suffix,
            parm.stack[ch].post_dir,
            parm.stack[ch].addon_low,
            parm.stack[ch].addon_high);
   }
   else     /* not found */
   {
      printf(“\r\nAddress not found.\r\n”);
   }

   printf(“\r\n”);
   printf(“\r\n-----------------------------------------------”);
   puts(“\r\nPress X to Exit or any other key to continue...\r\n”);

   ch = getchar();

   if (ch == ‘X’ || ch == ‘x’)
      rtncode=1;
   else
      rtncode=0;

   return rtncode;
}
```

Appendix B - Interface Definition

```
#ifndef ZIP4_H                   /* avoid redefinition */
#define ZIP4_H
#define RELVER          0xabcdef03L

/**************************************************************************/
/*   This record describes an address record. The record format is the  */
/*   the same as the USPS ZIP+4 File. Please see the USPS Address       */
/*   Information Products Technical Guide for information on this       */
/*   record.                                                            */
/*   NOTE:  All ‘char’ array fields contain an extra byte (+1) for      */
/*   the null terminator.                                               */
/**************************************************************************/
typedef struct
{
   char     detail_code;         /* copyright detail code            */
   char     zip_code[5+1];       /* zip code                         */
   char     update_key[10+1];    /* update key number                */
   char     action_code;         /* action code                      */
   char     rec_type;            /* record type                      */
   char     carr_rt[4+1];        /* carrier route                    */
   char     pre_dir[2+1];        /* pre-direction abbrev             */
   char     str_name[28+1];      /* street name                      */
   char     suffix[4+1];         /* suffix abbrev                    */
   char     post_dir[2+1];       /* post-direction abbrev            */
   char     prim_low[10+1];      /* primary low range                */
   char     prim_high[10+1];     /* primary high range               */
   char     prim_code;           /* primary even odd code            */
   char     sec_name[40+1];      /* bldg or firm name                */
   char     unit[4+1];           /* secondary abbreviation           */
   char     sec_low[8+1];        /* secondary low range              */
   char     sec_high[8+1];       /* secondary high range             */
   char     sec_code;            /* secondary even odd code          */
   char     addon_low[4+1];      /* add on low                       */
   char     addon_high[4+1];     /* add on high                      */
   char     base_alt_code;       /* base alternate code              */
   char     lacs_status;         /* LACS converted status            */
   char     finance[6+1];        /* finance code                     */
   char     state_abbrev[2+1];   /* state abbreviation  (not filled) */
   char     county_no[3+1];      /* county number                    */
   char     congress_dist[2+1];  /* congressional district           */
   char     municipality[6+1];   /* municip. city/state key (not filled) */
   char     urbanization[6+1];   /* urb. city/state key              */
   char     last_line[6+1];      /* last line city/state key         */
} ADDR_REC;

/* NOTE: The GovtBldgInd (Government Building Indicator) field is not */
/* available in the ADDR_REC structure. */

/*****************************************************************************/
/*   This record describes a city/state record.  The record format is the  */
/*   same as the USPS City State File.  Please see the USPS Address        */
/*   Infomation Products Technical Guide for information on this record.   */
/*   NOTE: All 'char' array fields contain an extra byte (+1) for the      */
/*          null terminator.                                               */
/***************************************************************************/
typedef struct
{
   char        detail_code;         /* copyright detail code          */
   char        zip_code[5+1];       /* zip code                       */
   char        city_key[6+1];       /* city/state key                 */
   char        zip_class_code;      /* zip classification code        */
                                    /* blank = non-unique zip         */
                                    /* M=APO/FPO military zip         */
                                    /* P=PO BOX zip                   */
                                    /* U=Unique zip                   */
   char        city_name[28+1];     /* city/state name                */
   char        city_abbrev[13+1];   /* city/state name abbrev         */
   char        facility_cd;         /* facility code                  */
                                    /* A=Airport mail facility        */
                                    /* B=Branch                       */
                                    /* C=Community post office        */
                                    /* D=Area distrib. center         */
                                    /* E=Sect. center facility        */
                                    /* F=General distrib. center      */
                                    /* G=General mail facility        */
                                    /* K=Bulk mail center             */
                                    /* M=Money order unit             */
                                    /* N=Non-postal name              */
                                    /*   community name,              */
                                    /*   former postal facility,      */
                                    /*   or place name                */
                                    /* P=Post office                  */
                                    /* S=Station                      */
                                    /* U=Urbanization                 */
   char        mailing_name_ind;    /* mailing name indicator         */
                                    /* Y=Mailing name                 */
                                    /* N=Non-mailing name             */
   char        last_line_num[6+1];  /* preferred last line key        */
   char        last_line_name[28+1];/* preferred city name            */
   char        city_delv_ind;       /* city delivery indicator        */
                                    /* Y=Office has city              */
                                    /*   delivery carrier rts         */
                                    /* N=Office does not have         */
                                    /*   city delivery carrier        */
                                    /*   routes                       */
   char        auto_zone_ind;       /* automated zone indicator       */
                                    /* Y=Automated facility           */
                                    /* N=Not an automated             */
                                    /*   facility                     */
   char        unique_zip_ind;      /* unique zip name indicator      */
                                    /* Y=Unique zip name              */
                                    /* blank=not applicable           */
   char        finance[6+1];        /* finance code                   */
   char        state_abbrev[2+1];   /* state abbreviation             */
   char        county_no[3+1];      /* county number                  */
   char        county_name[25+1];   /* county name                   */
} CITY_REC;

/**************************************************************************/
/*   Parameter list for z4adrinq() and z4xrfinq() calls.  Reserved         */
/*   fields are for future use, do not access these fields.  Size of this  */
/*   record cannot be changed.                                             */
/*   NOTE:  Only fields containing +1 in the length are null terminated.   */
/**************************************************************************/
typedef struct
{
                                    /*********** input data ***********/
   long        relver;              /* release version               */
   char        iadl1[50+1];         /* input delivery address        */
   char        iadl2[50+1];         /* input firm name               */
   char        ictyi[50+1];         /* input city                    */
   char        istai[2+1];          /* input state                   */
   char        izipc[10+1];         /* input ZIP+4 code              */
   char        iprurb[28+1];        /* input urbanization name       */
   char        iad13[50+1];         /* input second address line     */
   char        rsvd1[98];           /* reserved for future use       */

                                    /********** returned data *********/
   char        dadl3[50+1];         /* standardized 2nd delivery address*/
   char        dadl1[50+1];         /* standardized delivery address  */
   char        dadl2[50+1];         /* standardized firm name        */
   char        dlast[50+1];         /* standardized city/state/zip   */
   char        dprurb[28+1];        /* output PR urbanization name   */
   char        dctys[28+1];         /* main post office city         */
   char        dstas[2+1];          /* main post office state        */
   char        dctya[28+1];         /* standardized city             */
   char        abcty[13+1];         /* standardized city abbreviation */
   char        dstaa[2+1];          /* standardized state            */
   char        zipc[5+1];           /* 5-digit zip code              */
   char        addon[4+1];          /* ZIP+4 addon code              */
   char        dpbc[3+1];           /* delivery point bar code       */
   char        cris[4+1];           /* carrier route                 */
   char        county[3+1];         /* FIPS county code              */
   short       respn;               /* number of returned responses  */
   char        retcc;               /* return code                   */
   char        adrkey[12];          /* address key (for indexing)    */
   char        auto_zone_ind;       /* Y or N                        */
   char        lot_num[4+1];        /* LOT Number                    */
   char        lot_code;            /* LOT Ascending/Descending Flag  */


                                    /******* parsed input data ********/
   char        ppnum[10+1];         /* Primary Number                */
   char        psnum[8+1];          /* Secondary Number              */
   char        prote[3+1];          /* Rural Route Number            */
   char        punit[4+1];          /* Secondary Number Unit         */
   char        ppre1[2+1];          /* First or Left Pre-direction   */
   char        ppre2[2+1];          /* Second or Right Pre-direction  */
   char        psuf1[4+1];          /* First or Left Suffix           */
   char        psuf2[4+1];          /* Second or Right Suffix         */
   char        ppst1[2+1];          /* First or Left Post-direction   */
   char        ppst2[2+1];          /* Second or Right Post-direction */
   char        ppnam[28+1];         /* Primary Name                   */

   char        mpnum[10+1];         /* Matched primary number.        */
   char        msnum[8+1];          /* Matched secondary number       */
   char        pmb[3+1];            /* PMB Unit Designator            */
   char        pmbnum[8+1];         /* PMB Number                     */
   char        rsvd2[99];           /* reserved for future use        */


   struct {                         /************ footnotes ***********/
   char  a;                         /* zip corrected                  */
   char  b;                         /* city/state corrected           */
   char  c;                         /* invalid city/state/zip         */
   char  d;                         /* no zip assigned                */
   char  e;                         /* ZIP assigned for mult response */
   char  f;                         /* no zip available               */
   char  g;                         /* part of firm moved to address  */
   char  h;                         /* secondary number missing       */
   char  i;                         /* insufficient/incorrect data    */
   char  j;                         /* dual input – used PO BOX       */
   char  k;                         /* dual input – used non-PO BOX   */
   char  l;                         /* del addr component add/del/chg */
   char  m;                         /* street name spelling changed   */
   char  n;                         /* delivery addr was standardized */
   char  o;                         /* reserved for future use        */
   char  p;                         /* better delivery addr exists    */
   char  q;                         /* reserved for future use        */
   char  r;                         /* reserved for future use        */
   char  s;                         /* invalid secondary number       */
   char  t;                         /* reserved for future use        */
   char  u;                         /* unofficial PO name             */
   char  v;                         /* unverifiable city/state        */
   char  w;                         /* small town default             */
   char  x;                         /* unique ZIP Code default        */
   char  y;                         /* reserved for future use        */
   char  z;                         /* ZIP Move Match                 */
   char  rsvd3[6];                  /* reserved for future use        */
   } foot;

   ADDR_REC stack[10];              /********** record stack **********/
   char     rsvd4[194];             /* reserved for future use        */
} ZIP4_PARM;

/****************************************************************************/
/*   Parameter list for z4getzip()                                         */
/*   NOTE:  Only fields containing +1 in the length are null terminated.   */
/****************************************************************************/
typedef struct
{
    char    output_cityst[50+1];
    char    low_zipcode[5+1];
    char    high_zipcode[5+1];
    char    finance_num[6+1];
} GET_ZIPCODE_STRUCT;
/***************************************************************************/
/*   Return codes for z4adrinq() and z4xrfinq() calls.                    */
/***************************************************************************/
#define Z4_INVADDR  10              /* invalid address                 */
#define Z4_INVZIP   11              /* invalid zip code                */
#define Z4_INVSTATE 12              /* invalid state code              */
#define Z4_INVCITY  13              /* invalid city                    */
#define Z4_NOTFND   21              /* address not found               */
#define Z4_MULTIPLE 22              /* multiple response - no default */
#define Z4_SINGLE   31              /* single response - exact match  */
#define Z4_DEFAULT  32              /* default response                */

/****************************************************************************/
/* Parameter list for z4opencfg()                                           */
/* NOTE: Only fields containing +1 in the length are null terminated.       */
/****************************************************************************/

/*Use of this structure will replace a physical cop of the configuration    */
/*file on the hard drive                                                     */

typedef struct
{
   char *address1;     /*Contains the full path of the ZADRFLE.DAT file     */
   char *addrindex;    /*Contains the full path of the ZADRFLE.NDX file     */
   char *cdrom;        /*Contains the drive letter of the CD-ROM drive that */
                       /*contains the ZIP+4/carrier route data;may be blank */
   char *citystate;    /*Contains the full path of the following files:     */
                       /*CTYSTATE.DAT   - CTYSTATE.NDX                      */
                       /*ZIP5FILE.DAT   - ZIP5FLE.NDX                       */
   char *crossref;     /*Contains the full path of the ZXREFDTL.DAT file    */
   char *system;       /*Contain the full path of the Z4CXLOG.DAT file      */
}CONFIG_PARM;

typedef struct
{
   char  rsvd1[50];    /*reserved for future use                            */
   short status;       /*1 - Used value point to by fname                   */
                       /*2 - Used values in CONFIG_PARM                     */
                       /*9 - No values found. Search for z4config.dat       */
   char  *fname;       /*pointer to a NULL terminated string that           */
                       /*contains the full path and filename for a custom   */
                       /*config file. If fname contains a leading space     */
                       /*or NULL then it is ignored and the CONFIG_PARM     */
                       /*is evaluated for path names                        */
   CONFIG_PARM config  /*Contains the path name for the config file        */
   char  lotflag;      /*Y Enables LOT else Disalbe LOT                    */
   char  rsvd2[50];    /*reserved for future use                           */

}Z4OPEN_PARM
```

```
/****************************************************************************/
/*              Z4OPEN_PARM.status values for z4opencfg(1)                  */
/****************************************************************************/

#define Z4_FNAME     1  /*Used the value in fname as the path and filename  */
#define Z4_CONFIG    2  /*Used the paths in the CONFIG_PARM structure       */
#define Z4_SEARCH    9  /*Used neither, searched for z4config.dat           */

/***************************************************************************/
/*   Function prototypes for the ZIP+4 retrieval engine.                   */
/***************************************************************************/
#if defined(OS2_32)
#define Z4FUNC
#elif defined(WIN32)
#define Z4FUNC _cdecl
#elif defined(_WINDOWS) || defined(_WINDLL)
#define Z4FUNC __far __pascal __export
#elif defined(OS2)
#define Z4FUNC _far _pascal _loadds _export
#elif defined(_MAC)
#define Z4FUNC
#elif defined(ANSI_STRICT) || defined(UNIX) || defined(I370)
#define Z4FUNC
#else
#define Z4FUNC _cdecl
#endif

int  Z4FUNC z4ready(void);           /* check presence of retrieval engine*/
int  Z4FUNC z4remove(void);          /* terminate the retrieval engine   */
int  Z4FUNC z4open(void);            /* open the retrieval engine for use */
int  Z4FUNC z4close(void);           /* close the retrieval engine       */
int  Z4FUNC z4abort(void);           /* abort the current inquiry        */
int  Z4FUNC z4adrinq(ZIP4_PARM *);   /* address inquiry                  */
int  Z4FUNC z4scroll(ZIP4_PARM *);   /* address inquiry                  */
int  Z4FUNC z4adrkey(ZIP4_PARM *);   /* address key (for indexing)       */
int  Z4FUNC z4xrfinq(ZIP4_PARM *);   /* nine digit cross reference inquiry*/
int  Z4FUNC z4adrstd(ZIP4_PARM *, int);/* address standardization        */
int  Z4FUNC z4ctyget(CITY_REC *, void *);/* get first city for a state   */
int  Z4FUNC z4ctynxt(CITY_REC *);    /* get next city for a state        */
int  Z4FUNC z4adrget(ADDR_REC *, void *);/* get first address for a finance no   */
int  Z4FUNC z4adrnxt(ADDR_REC *);    /* get next address for a finance no */
int  Z4FUNC z4date(char *);          /* get date of ZIP+4 database       */
int  Z4FUNC z4expire(void);          /* number of days until expiration  */
int  Z4FUNC z4getzip(GET_ZIPCODE_STRUCT *);/* get zip code range for cityst    */

#endif /* ZIP4_H */
```

Appendix C – File Names and Locations

Z4CONFIG.DAT

Data file used to specify the location of the Address Matching System data files. You are responsible for creating this file, or you may use the skeleton file provided with the developer's kit. This file should be located in the current working directory of the application using the API.

SAMPLE Z4CONFIG.DAT

```
APPLICATION     OTHER - ZIP+4
COMPUTER        OTHER
ADDRESS1        D:\AMSDATA\
ADDRESS2
ADDRESS3
ADDRINDEX       D:\AMSDATA\
CDROM
CITYSTATE       D:\AMSDATA\
CROSSREF        D:\AMSDATA\
SYSTEM          C:\AMS_KIT\
TABLE
USER
ADDR1SIZE
ADDR2SIZE
ADDR3SIZE
```

Depending on the capacity of your hard drive, copy the files identified in the next step from the CD-ROM. Directory and drive listings should be entered according to the specific computer platform being used. Each directory listing **must** contain a trailing directory delimiter. See the Section 2, Installation Procedures, for a sample Z4CONFIG.DAT listing for each supported computer platform.

SAMPLE Z4CONFIG.DAT LINE DESCRIPTION

Sample File Line	Description
APPLICATION	Not used by the Address Matching System API
COMPUTER	Not used by the Address Matching System API
ADDRESS1	Contains the full path of the ZADRFLE.DAT file
ADDRESS2	Must be present and empty
ADDRESS3	Must be present and empty
ADDRINDEX	Contains the full path of the ZADRFLE.NDX file
CDROM	Contains the drive letter of the CD-ROM drive that contains the ZIP+4/ carrier route data; this listing may be blank
CITYSTATE	Contains the full path of the following files: CTYSTATE.DAT CTYSTATE.NDX ZIP5FLE.DAT

	ZIP5FLE.NDX
CROSSREF	Contains the full path of the following files:
	ZXREFDTL.DAT
	LTRVFLE.DAT
	LTRVFLE.NDX
SYSTEM	Contains the full path of the Z4CXLOG.DAT file
TABLE	Not used by the Address Matching System API
USER	Not used by the Address Matching System API
ADDR1SIZE	Not used by the Address Matching System API
ADDR2SIZE	Not used by the Address Matching System API
ADDR3SIZE	Not used by the Address Matching System API

Note: *If you change the location of any of the files in this list, you must also change the corresponding path in your Z4CONFIG.DAT.*